Attention

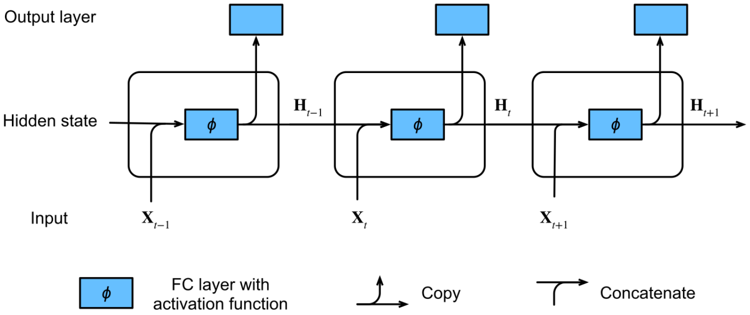
HE Jiayou

July 7, 2022

Table of Contents

Outline

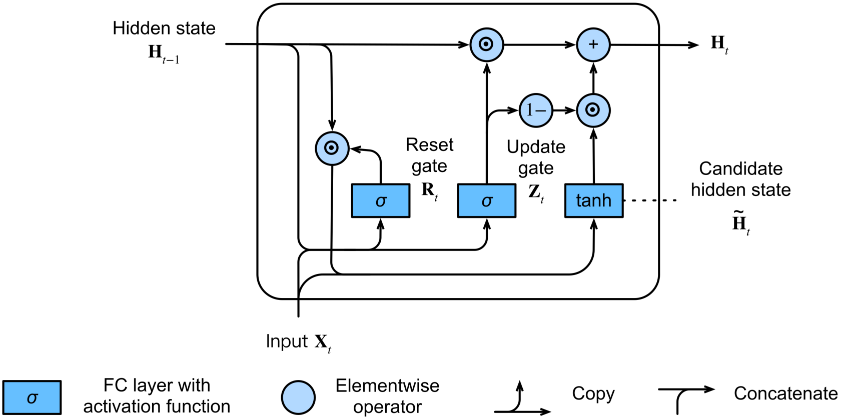
RNN



$$H_t = \phi\left(X_t W_{xh} + H_{t-1} W_{hh} + b_h\right)$$
$$O_t = H_t W_{hq} + b_q$$

$$X_t \in \mathbb{R}^{n\times d} \quad H_t \in \mathbb{R}^{n\times h}$$
$$O_t \in \mathbb{R}^{n\times q} \quad b_h \in \mathbb{R}^{1\times h}$$

GRU Gated Recurrent Unit



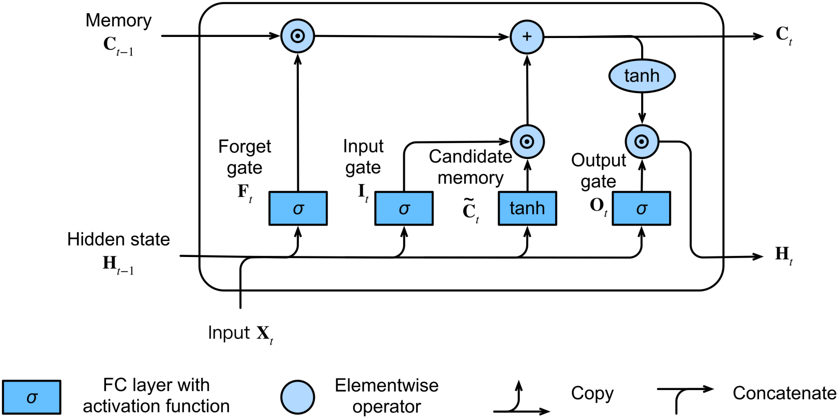
GRU Gated Recurrent Unit

GRU supports gating of the hidden state.

- Reset gates help capture short-term dependencies in sequences.
- Update gates help capture long-term dependencies in sequences.

$$
\begin{aligned}
R_t &= \sigma\left(X_t W_{xr} + H_{t-1} W_{hr} + b_r\right) \\
Z_t &= \sigma\left(X_t W_{xz} + H_{t-1} W_{hz} + b_z\right) \\
\tilde{H}_t &= \tanh\left(X_t W_{xh} + \left(R_t \odot H_{t-1}\right) W_{hh} + b_h\right) \\
H_t &= Z_t \odot H_{t-1} + \left(1 - Z_t\right) \odot \tilde{H}_t
\end{aligned}
$$

LSTM



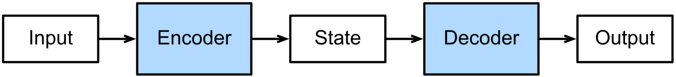
Memory
$\mathbf{C}_{t-1}$
$\mathbf{C}_t$
+
tanh
Forget
gate
$\mathbf{F}_t$
Input
gate
$\mathbf{I}_t$
Candidate
memory
$\tilde{\mathbf{C}}_t$
Output
gate
$\mathbf{O}_t$
σ
tanh
Hidden state
$\mathbf{H}_{t-1}$
$\mathbf{H}_t$
Input $\mathbf{X}_t$
FC layer with
activation function
Elementwise
operator
Copy
Concatenate

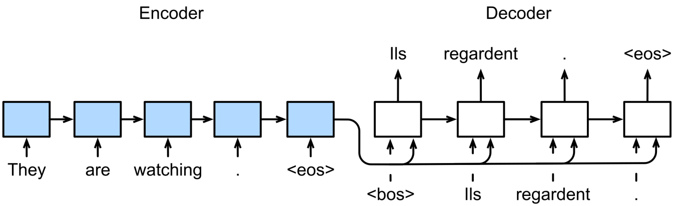
LSTM

The idea is similar to GRU.

$$
\begin{aligned}
I_t &= \sigma\big(X_t W_{xi} + H_{t-1} W_{hi} + b_i\big)\\
F_t &= \sigma\big(X_t W_{xf} + H_{t-1} W_{hf} + b_f\big)\\
O_t &= \sigma\big(X_t W_{xo} + H_{t-1} W_{ho} + b_o\big)\\
\tilde{C}_t &= \tanh\big(X_t W_{xc} + H_{t-1} W_{hc} + b_c\big)\\
C_t &= F_t \odot C_{t-1} + I_t \odot \tilde{C}_t\\
H_t &= O_t \odot \tanh(C_t)
\end{aligned}
$$

Encoder-Decoder

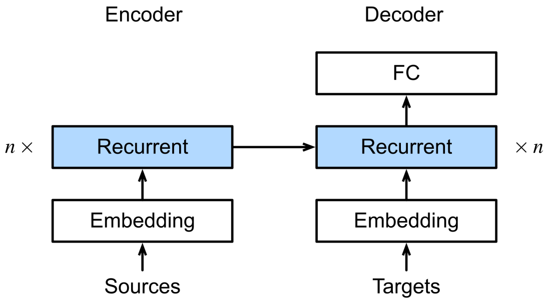




For the purpose of *variable* input and output sequences.

Encoder-Decoder

- Encoder: $H_t = f\big(X_t, H_{t-1}\big),\ C = g\big(H_1, \ldots, H_t\big)$
- Decoder: to get $P(Y_t | Y_1, \ldots, Y_{t-1}, C),\ H_t = g\big(Y_{t-1}, C, H_{t-1}\big)$.



Outline

Attention Prompt

A simple regression Problem: $f \in \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$.

- Average Pooling:

$$f(x) = \frac{1}{n} \sum_{i=1}^{n} y_i$$

- Attention Pooling:

$$f(x) = \sum_{i=1}^{n} \alpha(x, x_i) y_i$$

We call x a *query* and (x_i, y_i) a *key-value* pair.
α is the attention weight, which is the target.

Attention Prompt

- Nonparametric:

$$\alpha(x, x_i) = \frac{K(x - x_i)}{\sum_j K(x - x_j)}$$

$$K(u) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{u^2}{2}\right)$$

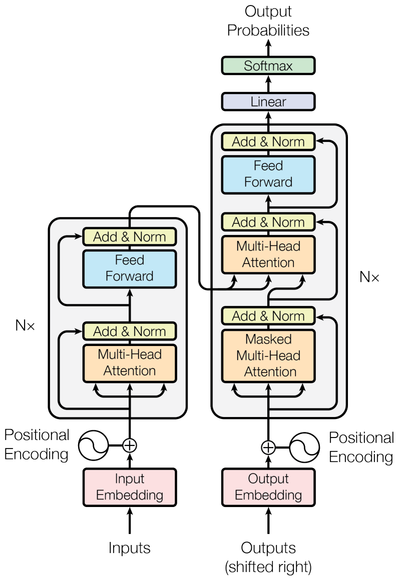
- parametric: learnable *Attention Scoring Function*

Outline

- Relys entirely on self-attention
- Encoder-Decoder architecture
- Positional encoding

1

[1] **Attention**.



Encoder:

- $N = 6$ layers
- Multi-head self-attention + feed forward

Decoder:

- Masked Multi-head self-attention
- Multi-head attention

Others:

- Positional Encoding
- Layer-normalization

Scoring Function

Scaled Dot-Product Attention

$$Attention(Q, K, V) = \mathsf{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$Q \in \mathbb{R}^{n \times d_k} \quad K \in \mathbb{R}^{m \times d_k} \quad V \in \mathbb{R}^{m \times d_v}$$

Multi-Head Attention

It is beneficial to linear project Q, K, V to d_k, d_k, d_v dimensions h times.

$$Multihead(Q, K, V) = Concat(head_1, \dots, head_n)\, W^O$$
$$\text{where } head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

Self-Attention

Self-attention:

$$\{f(x_i), 1 \leq i \leq n\}$$
$$\text{where } f \in \{(x_i, x_i)\}$$

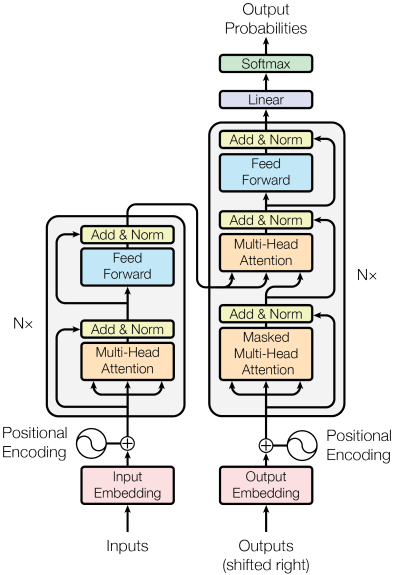
Positional Encoding

For $P \in \mathbb{R}^{n \times d}$:

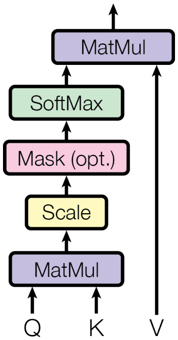
$$p_{pos,2i} = \sin\left(\frac{i}{10000^{2i/d}}\right)$$
$$p_{pos,2i+1} = \cos\left(\frac{i}{10000^{2i/d}}\right)$$

n length of sequence; d length of encoding.
Give each position-embedding pair a *unique* value.

Recap



Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Add & Norm
Masked
Multi-Head
Attention
Nx
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Positional
Encoding
Input
Embedding
Output
Embedding
Inputs
Outputs
(shifted right)

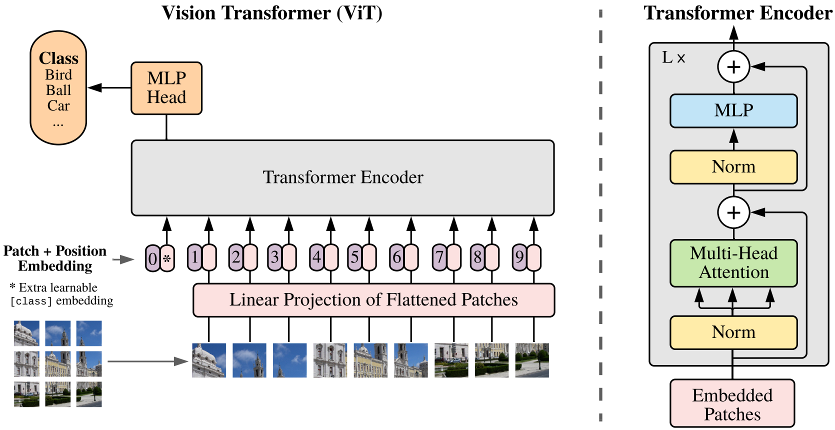


MatMul
SoftMax
Mask (opt.)
Scale
MatMul
Q
K
V

Outline

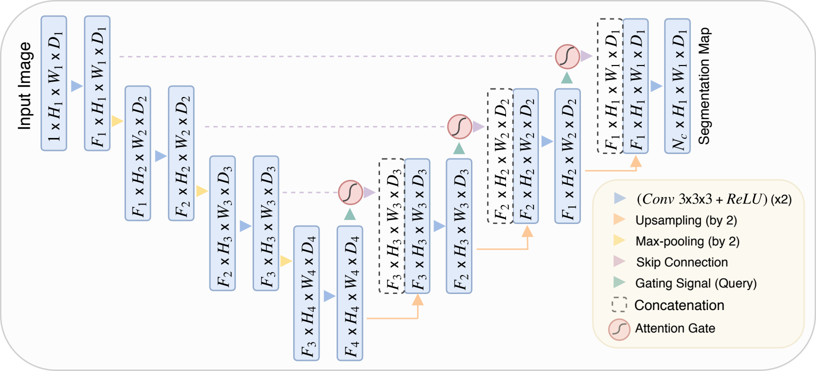
VIT Vision Transformer

Split images into fixed-size patches



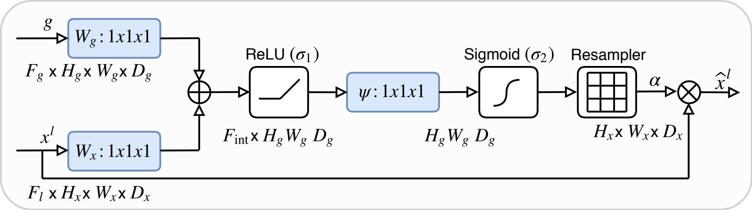
Outline

Attention UNet



$$Concat(Attention, upsample)$$

Attention Unet



Using *query* g from a coarser scale to provide attention scoring function.
Trilinear interpolation is applied.

Outline

Inductive Bias

Component	**Entities**	**Relations**	**Rel. inductive bias**	**Invariance**
Fully connected	Units	All-to-all	Weak	-
Convolutional	Grid elements	Local	Locality	Spatial translation
Recurrent	Timesteps	Sequential	Sequentiality	Time translation
Graph network	Nodes	Edges	Arbitrary	Node, edge permutations

From GNN. SS